METHOD OF REPAIRING INTEGRATED CIRCUITS

1. Background of the invention

1.1. Related art

1.1.1. Multidimensional arrays of arithmetic and logic units

5   German Patent 196 51 075.9-53 describes processors having a plurality of 2-dimensional or multidimensional arithmetic and logic units/cells. The computing power of such processors increases with the number of arithmetic and logic units present. Therefore, an attempt is made to integrate as many

10   arithmetic and logic units as possible on one chip, which increases the area required. With an increase in area, there is also a higher probability of a chip having a manufacturing defect making it useless. All arithmetic and logic units arranged in matrix form have this problem, e.g., including

15   other known types such as DPGAs, Kress arrays, systolic processors and RAW machines; likewise, some digital signal processors (DSPs) having more than one arithmetic and logic unit.

20   At the same time, all the aforementioned types require a great deal of testing, i.e., to detect faults, an especially large number of test cases must be generated and tested with respect to the functioning of the cells and the networking. Traditional known methods such as BIST, boundary scan, etc.

25   are difficult to integrate because of the large number of test vectors and they are also too time consuming and take up too much space.

1.1.2. Standard processors

Standard processors such as the known x86 series, MIPS or ALPHA have a plurality of arithmetic and logic units which are driven at the same time by a VLIW command or with a time offset. In the future, the number of integrated units (integer units) and floating point units will continue to increase. Each unit must be tested adequately and must be largely free of defects.

1.2. Problems
1.2.1. Multidimensional arrays of arithmetic and logic units

Due to the increasing probability of defects with large chips, either only a very small number of cells can be integrated or production costs will increase greatly due to the resulting rejects. Very large chips will reach a maximum area beyond which a functional chip can no longer be produced. Due to the time consumed in testing according to traditional methods, there is a great increase in testing costs. Integrated BIST functions (built-in self-test) take up a great deal of area due to the high extra complexity, driving costs even higher and reducing manufacturing feasibility. In addition, this greatly increases the probability of a defect lying not within the actual function units but instead within the test structures.

1.2.2. Standard processors

Due to the increasing number of arithmetic and logic units, there is also an increase in the probability of defects. This means more rejects, causing manufacturing costs to increase. With an increase in area and a related increase in the number of transistors used, there is also an increase in probability of failure during use.

With regard to testing complexity and implementation of BIST,

the discussion above regarding "multidimensional arrays of arithmetic and logic units" also applies here.

1.3. Improvement through the present invention; object

According to the present invention, it is possible to replace defective cells by functional cells and thus reduce rejects. A cell can be replaced either by the test systems at the time of manufacture of the chips or even by the user in the completely assembled system. Test vectors can be generated according to the BIST principle within the chip, or outside the unit according to a new method to save on space and costs. In addition, a possibility of chips automatically repairing defects without requiring any additional external tool is described. All the tests and repairs can be performed during operation of the chips.

2. Description of the present invention
2.1. Detailed description of the present invention
2.1.1. Replacing defective cells

An additional PAE not used in normal operation (referred to below as PAER) is assigned to a group of cells which are referred to below as PAEs according to German Patent 196 51 075.9-53. The cells may be arithmetic and logic units of any type, configurable (programmable) logic cells or other cores having any desired function. Grouping of the PAEs in rows or columns is preferred in the grouping of PAEs and allocation of the PAER, because this simplifies the networking. With respect to future chip technologies, reference is made to a possible grouping of the PAEs within a 3rd dimension. Multiplexers are connected upstream from the inputs of the PAEs in such a way that the input of the first PAE in the row/column can also be switched to the input of the second PAE in the row/column, and then the input of the second PAE can be switched to the input of the third PAE and so forth. The input of the last PAE is switched to the input of the PAER. This means that if there is

a defect in the first PAE, its function is replaced by the second PAE, the function of the second is replaced by the third and so forth, until the function of the last PAE is replaced by the PAER. If a PAE within the column/row is defective, the PAEs upstream from it are switched normally and after the position of the defective PAE, all functions are shifted by one PAE. For example, if PAE 4 is defective, then PAEs 1...3 execute their respective functions, while the input multiplexer of PAE 5 is switched so that it receives the data of PAE 4, the input multiplexer of PAE 6 receives the data of PAE 5 and so forth until the input of the PAER receives the data of the last PAE.

To supply the results back to the network in the proper sequence, multiplexers are also provided at the outputs of the PAEs, with the output multiplexer of PAE 1 either switching PAE 1 to the bus (if it is not defective) or if there is a defect, switching the output of PAE 2 to the bus, PAE 3 is switched to the bus instead of PAE 2, until the last PAE, where the PAER is switched in its place. If the defective PAE is in the middle of the row/column, the outputs are shifted exactly as already described above for the inputs.

Especially with a configurable logic and configurable arithmetic and logic units, there are additional bus systems to transfer the configuration data and control the configuration. These bus systems are also connected by multiplexers in the same way as the buses mentioned in this section. The same thing is also true of bus systems over which commands are written to the respective arithmetic and logic units with a matrix arrangement of arithmetic and logic units (e.g., systolic processors, SIMD, etc.). Basically any bus or any signal can be sent over multiplexers. Depending on the fault tolerance requirements, the clock signal, for example, can be sent over multiplexers to prevent a possible short circuit, or the clock signal may be sent directly to the cell because such a failure need not be compensated. The fault

tolerance step can be defined in the structural details according to the requirements for each signal or each bus individually.

5   2.1.2. Replacing defective buses

The concept of correcting faults within gate structures as presented here can also be applied accordingly to bus systems where an additional bus (BUS R) is assigned to a number of
10   buses (BUS 1 ... BUS n). If one of the buses is defective (BUS d), its function is assumed by one of its neighboring buses (BUS (d + 1)). The function of the neighboring bus (BUS (d + 1)) is assumed by its neighboring bus (BUS (d + 2)), etc., with the direction of the bus assuming the function always
15   remaining the same until the function of BUS n is assumed by BUS R.

When multiplexer structures are used with bus systems, the usual multiplexers, decoders and gates, tristate gates or
20   bidirectional multiplexers are used according to the prevailing connection structure and the direction of the data.

2.1.3. Decoder

25   It is obvious that two groups of successive multiplexers must always assume the same state, i.e., MUX 1 = MUX 2 = MUX 3 = ... = MUX n = state A, and MUX (n + 1) = MUX (n + 2) = MUX (n + 3) = ... = MUX m = state B.

30   If no PAE is defective, then MUX 1 = MUX 2 = ... = MUX m = state A.

If the first PAE is defective, then MUX 1 = MUX 2 = ... = MUX m = state B.
35

For example, if PAE 3 is defective, then MUX 1 = MUX 2 = state A, MUX 3 = MUX 4 = ... = MUX m = state B, with PAER being

assigned to PAE m in this example, i.e., PAER is directly next to PAE m.

The multiplexers are therefore controlled as follows, for example:

Defective PAE	Multiplexer control
none	0000...000
m	0000...001
m-1	0000...011
m-2	0000...111
4	0001...111
3	0011...111
2	0111...111
1	1111...111

If PAER is assigned to PAE 1, the m...1 sequence is inverted (defective PAE 1 corresponds to 0000...001, or defective PAE m corresponds to 1111...111).

Therefore, it is sufficient to store the number of the defective PAE and send it to a decoder which controls the states of the multiplexers on the basis of the table given above.

2.1.4. Performing the self-test

Essentially any desired test strategies can be applied to this method, but the following method according to the present invention can be regarded as especially suitable.

The array of PAEs is loaded with one or more test algorithms which calculate one or more test vectors. At one edge of the array, the PAEs are wired as comparators so that the values calculated on the basis of the test vectors are compared to the setpoint results.

If the calculated result does not correspond to the setpoint results, there is a defect. The test data, i.e., the test algorithms, the test vectors and the setpoint results are present in an internal or external memory or are loaded by a higher level unit. In this test strategy, it is necessary for each test algorithm to be calculated at least twice, with the PAEs designed as comparators being on another edge (preferably the opposite edge) the second time to guarantee execution of the test algorithm on all PAEs. It is also conceivable for the comparators to be arranged in the interior of the PAE array, and with one test algorithm A and B each calculating one result A and B from left and right (top and bottom), where the results are sent to the comparators and must match. Depending on the type of test algorithm, the defective PAE may be tracked back on the basis of the defect or not. If the algorithm supports tracking of the defect, the corresponding multiplexer states of the row/column in which the defective PAE is located are changed and sent to the multiplexers. The test algorithm in which the test is failed is executed again to check on freedom from defects, which should now prevail. If the unit is still defective, a check must be performed to determine whether an additional or other PAEs are defective. Execution of the test algorithm and generation of the multiplexer states adapted to the defect are iterative processes. It is not usually sufficient to implement just one test algorithm, but instead multiple different test algorithms must be implemented, each being checked with multiple test vectors. Only in this way can a maximum fault detection rate be achieved.

At the same time, the bus crosslinking must be changed from one test algorithm to the next, so that the bus systems are also checked adequately.

The various embodiments of the test algorithms will not be discussed in further detail here because this is not relevant for the basic method according to the present invention.

2.1.5. Internal control of the self-test

Units such as German Patent 196 51 075.9-53, DPGAs, Kress arrays, systolic processors and RAW machines all have in common an integrated memory which is assigned to one or more PAEs and determines the function of the arithmetic and logic unit(s).

According to the basic BIST principle, the memory is expanded by a region (TestMEM) containing the test algorithms and vectors. This memory can be fixed in the form of a ROM or it may be rewritable by (E)EPROM, flash ROM, NV-RAM or the like.

To perform a self-test, the system jumps to a memory location within the TestMEM and executes the test routine stored there (internal driven self-test = IDST). Except for the expansion of the memory (by the TestMEM) and an analyzer unit for the comparators described above (ErrorCHK), no other additional units typical of BIST are needed on the chip.

2.1.6. External control of the self-test

The reduction in components on a memory expansion (TestMEM) and an analyzer unit of the comparators (ErrorCHK) permit an additional, even less expensive and space-saving variant. No internal TestMEM is implemented here, but instead the usual internal memory is loaded from the outside with the test algorithm and the test vectors (external driven self-test = EDST); this means that the BIST test data is shifted outward and regarded as a normal program. Then the test algorithm is executed. As an alternative, the test algorithm may also be loaded successively from an external memory during the execution and decoded. Only the ErrorCHK unit must still be integrated on the chip. There are several possibilities for loading the test algorithm and the test vectors from the outside into the chip-internal memory (memories). In principle, the process can take place through a functionally

higher level CPU or computer unit (HOST), with the latter
loading the test data (test algorithm and test vectors) onto
the chip (download) or the chip loading the test data
automatically from an external (dual-ported) RAM or read-only

5   memory such as ROM, (E)EPROM, flash ROM, NV-ROM or the like.

2.1.7. Checking the function during operation

BIST methods according to the related art usually perform the

10   self-test only during the chip RESET phase, i.e., shortly
after applying a voltage (when turned on). In contrast with
that, it is possible or practical to perform the methods
described here on the chips while the programs are running.
For example, a complete test of the chip can be performed

15   during the RESET phase and part of the available test data can
be loaded during execution of the application program or
during IDLE cycles, i.e., periods of time when no program is
running on the chips or the chip is in a waiting mode. This is
readily possible by the fact that one of the test algorithms

20   is activated in the internal memory during IDLE cycles or is
loaded into the unit from an external memory or host. It is of
course possible to select one or more of the plurality of
available test algorithms and test data in part, where the
number of selected test data can be defined on the basis of

25   the length of the IDLE cycle. New test data can be loaded
until the IDLE cycle is ended by the arrival of new data to be
processed, by a new program to be executed or by another
request.

30   Another possibility is the fixed integration of test
strategies into the application programs to perform tests
during processing of the application program. In both cases,
the relevant data in the array is saved before calling up the
test algorithms. One option is to save the data either in

35   internal memory areas (cf. PACT04) or in memories connected
externally. After executing the test algorithms, the data is
read back before the normal program processing.

One alternative for increasing the execution speed is to implement in addition to each register (Reg-n, n ∈ N) an additional register (TestReg-n, n ∈ N) which is used only for the test algorithms. Before execution of the test algorithms, the TestReg-n are connected by multiplexers/demultiplexers (gates) and used for the test. The Reg-n remain unchanged. After execution of the test algorithms, the Reg-n are connected again.

If the test strategy provides for testing of only those cells whose data is no longer relevant subsequently, the data need not be saved and loaded.

2.1.8. Storing the number of the defective PAE

If a PAE (or a bus) is recognized as defective, its number, i.e., the state vector (defect identifier) of the assigned multiplexer must be stored first to control the multiplexers and also to be available immediately for a chip RESET. Therefore, the defect identifier may be stored

1.   internally in the chip in a programmable read-only memory ((E)EPROM, flash ROM, NV-RAM, etc.),

2.   externally in a programmable read-only memory ((E)EPROM, flash ROM, NV-RAM, etc.),

3.   externally in the HOST within the program to be executed, in its programmable read-only memory ((E)EPROM, flash ROM, NV-RAM, etc.) or in other storage media (magnetic, optical, etc.).

2.1.9. Automatic generation of multiplexer states

Usually after detection of a defect, the defective cell is tracked on the basis of the defective performance detected. This is possible with appropriate test algorithms if there is an additional algorithm for tracking the defect. If the test is controlled by a HOST, the tracking can be performed on the HOST. However, if there is no HOST, tracking often cannot be

integrated into the defective chip or is too complicated. As
an expedient, it is proposed that a loadable counter be
integrated upstream from each decoder. In the normal case, the
number of the defective PAE is loaded into the counter, after

5   which the decoder controls the states of the multiplexers as
described above. If it is not known which PAE is defective,
beginning at PAE 0 or PAE m the counter can function in
response to any potentially defective PAE by reducing the
count by one PAE (counting from PAE m) or increasing the count

10   by one (counting from PAE 0) after each unsuccessful test
until the defective PAE is reached and the test takes place
normally. The count then reached is stored as the state vector
for controlling the multiplexers and represents the defective
PAE. If a functional count is not found, there is either

15   another defect (possibly in another row/column or a bus error)
or more than one PAE is defective. One disadvantage when using
counters is that all possibilities must be permutated until
the defective PAE has been located.

20   Another possibility, although it requires a greater
implementation expense, is therefore to use look-up tables
which select the corresponding defective PAE on the basis of
the test algorithm just executed and the resulting error state
in ErrorCHK. To do so, however, the test algorithms and look-

25   up tables must be coordinated. However, this coordination will
not be discussed further here because it is highly chip-
specific and does not depend on the basic principle.

2.1.10. Special design for standard processors (Pentium, MIPS,
30   ALPHA, etc.)

Processors today and those in the future will contain a
plurality of integer units and floating point units.
Therefore, the method described here can be applied directly
35   to these units by having an additional unit in each case which
will be available for possible defects. The test of the
processors can be performed at the manufacturer's, during

startup of the computer or also during the operating time. It is especially appropriate for a test to be performed during boot-up, i.e., starting up of the computer after a reset, which is performed with PCs of the BIOS type (BIOS = basic

5   input output system). The corresponding state vectors of the multiplexers can be stored either on the processor or in an external memory, e.g., the battery-buffered real-time clock (RTC) in a PC.

10   3. Summary

The present invention makes it possible to replace defective units, which are designed as arithmetic and logic units in the present publication but in general can represent any desired

15   unit of a chip, by a functional unit. At the same time, the present invention relates to a method with which self-tests can be performed more easily, less expensively and before or during the running of the application program. This also greatly increases fault tolerance during operation, which is

20   especially important for failure-critical applications such as power plant operations, aviation and space travel or in the military.

4. Brief description of the diagrams

25

The following diagrams illustrate embodiments of the method according to the present invention:

Figure 1:   basic circuit
Figure 2:   no PAE defective
30   Figure 3:   PAE1 defective
Figure 4:   PAEm defective
Figure 5:   PAE3 defective
Figure 6:   array of PAEs with PAERs
Figure 7:   fault-tolerant bus system
35   Figure 7a: principle of a self-test, first part
Figure 7b: principle of a self-test, second part
Figure 8a: EDST integrated into an external ROM

4.1. Detailed description of the diagrams

Figure 1 shows the basic principle of a fault-tolerant configuration. All the PAEs (0101) are arranged in a row with the additional PAER (0102) being assigned to the last PAE. Upstream from the first PAE in the row, a gate (0103) is connected to block the data to the PAE if that PAE is defective. Likewise, a gate (0105) is connected upstream from the PAER (0102) to block the data to the PAER if it is not needed (or is defective). The two gates (0103 and 0105) are optional and are not absolutely necessary. The input buses (0111) composed of a plurality of individual signals are directed over multiplexers (0104) and the gates (0103 and 0105) to the PAEs. After a defect, data can be shifted by one PAE to the right in each case up to the PAER. Upstream from

the output buses (0112), which are composed of a plurality of individual signals, there are also multiplexers (0106) which again shift the results by one position to the left in the event of a defect, so the defect is not detectable for the result bus system (quantity of all 0112). The individual control signals (0117) for the respective multiplexers and gates are combined to give one bus (0110) and are generated by a decoder (0107). The decoder receives the number of the defective PAE from unit 0108, which is designed either as a register or as a loadable counter. When using a look-up table to detect the defective PAE from the error generated by ErrorCHK, 0108 is implemented as a register into which the number of the defective PAE is loaded. If the defective PAE is sought by a permutation, 0108 represents a loadable counter which counts through all possible PAEs, starting from 0, until the defective PAE is located. Once the defective PAE has been identified, it is loaded directly into the loadable counter in the next RESET operation. The load signal LOAD (0115) is available for loading the counter or register (0108). To increment the counter, the COUNT signal (0116) is sent to the counter. The counter result is returned by way of signals 0114 for storage. The counter/register is driven and the chronological sequence is controlled by a state machine (not shown), an external HOST or a device according to German Patent 196 54 846.2.

Figure 2 shows the states of the multiplexers (0104 and 0106) as well as the gates (0103 and 0105) where no PAE is defective and the PAER (0102) is not used.

Figure 3 shows the states of the multiplexers (0104 and 0106) as well as the gates (0103 and 0105) where PAE 1 (0301) is defective and the PAER (0102) is used.

Figure 4 shows the states of the multiplexers (0104 and 0106) as well as the gates (0103 and 0105) where PAE m (0401) is defective and the PAER (0102) is used.

Figure 5 shows the states of the multiplexers (0104 and 0106) as well as the gates (0103 and 0105) where PAE 3 (0501) is defective and the PAER (0102) is used.

5   Figure 6 shows an array of PAEs (0601), where one PAER (0602) is assigned to each PAE row, and each row has a separate control (0109, see Figure 1). A plurality of controls can also be combined to a single higher-order control above the rows.

10   Figure 7a shows a possible first test of an algorithm, with a plurality of PAEs being configured as processor elements (0701), each performing its operation by using a test vector. The PAEs are interconnected by bus systems (0708) of any desired design. One row of PAEs (0702) is configured as a
15   comparator. The values calculated in the processor elements are compared to a predetermined value in the comparators. There is an error if the two values do not match. The results of the comparisons are sent over a bus system (0705) to a multiplexer (0703) which is connected so that it relays the
20   results of the comparisons from (0702) to a unit (ErrorCHK 0706) of any desired design to detect an error and possibly analyze it. The error analysis (0706) sends its result over the bus (0707) to the HOST or to the controlling state machine (see Figure 1).
25
Figure 7b shows a second test which is performed over the same matrix of PAEs as Figure 7b. The algorithm to be executed is the same; likewise the values to be calculated. However, the PAEs of row 0702 are designed as normal processor elements,
30   but with the first row of the PAEs (0701) connected previously as arithmetic and logic units (Figure 7a) now being designed as comparators (0711). The direction of data flow on the bus systems is rotated by 180°. The multiplexer (0703) is connected so that the results of the (mirrored) comparators
35   (0711) are relayed to the error analysis (0706). The mirroring of comparators (0702-0711) achieves the result that the function of each PAE is actually tested. If this mirroring

were not performed, only the function of comparison but not any desired function is tested in a PAE row (0702 or 0711).

Implementation of a method other than that on which Figures 7a and 7b are based may be appropriate under some circumstances. The matrix here is subdivided into three groups, an upper group, a middle group and a lower group. In the upper and lower groups, results are calculated, with the data flow of the upper group being downward and the data flow of the lower group being upward. The middle group is configured as a comparator and compares the calculated values of the upper group with those of the lower group. The upper and lower groups usually perform the same calculations. There is an error if the results obtained at the comparators are different. Again with this method, it is important for the PAEs which are connected as comparators to be tested adequately for freedom from defects in the next configuration.

Figures 8a, b, c show possible circuits for testing a chip (0801). The test data is stored externally here (EDST).

In Figure 8a the test data is in an external read-only memory ((E)PROM, ROM, flash ROM, etc.) (0802). The data is in a RAM (0808) through which it is exchanged with the HOST (0805). A non-volatile read-write memory (NV-RAM, EEPROM, flash ROM, etc.) (0807) is used to store the defective PAEs, i.e., the state vectors of the multiplexers.

In Figure 8b, the test data is loaded by a HOST (0805) into part of the RAM (0803) and executed from there. The data is in a RAM (0808) through which it is exchanged with the HOST (0805). It is also possible for the chip itself to load the data into the memory area (0803, 0808) (without the use of a HOST), e.g., directly from a bulk storage device. A non-volatile read-write memory (NV-RAM, EEPROM, flash ROM, etc.) (0807) is used to store the defective PAEs, i.e., the state vectors of the multiplexers.

The errors generated by the chip in Figures 8a/8b are brought out of the unit and are available externally (0804).

In Figure 8c the test data is transmitted incrementally from a
5   HOST (0805) to the chip (0801) using a suitable interface
(0806). The chip indicates a possible error state (0804) to
the HOST through the interface (0806). The data is in a RAM
(0808) through which it is exchanged with the HOST (0805).

10   Figure 9 illustrates an expansion of a chip-internal program
memory or configuration memory according to German Patent 196
54 846.2 by the addition of a BIST function according to the
method described here. The usual memory area (0901) according
to the related art is expanded here by a read-only memory area
15   (0902), which is usually implemented as ROM, but
implementation as (E)EPROM, flash ROM, NV-RAM, etc. in which
the test data, i.e., the test algorithms and test vectors are
stored would also be possible. At the end of the memory,
another memory area (0903) is added. This is a few entries in
20   size and includes the addresses of defective PAEs and/or
defective buses which are loaded into the counters/registers
(0108). This memory area is designed as a non-volatile read-
write memory (flash ROM, EEPROM, NV-RAM, etc.). Thus, the data
can be read out in a RESET and written over with the
25   instantaneous data after execution of a test algorithm where a
reparable defect is found. This data is supplied by the
counters (0108, bus 0114) or by the look-up tables, depending
on the implementation.

30   Figure 10 shows the example of a standard processor with fault
correction implemented. According to the related art, the
processor is connected to its peripherals over the bus
interface (1001). Two multiplexers (1002 and 1003) are
assigned to the bus interface, with 1002 controlling two
35   alternative code caches (1004, 1005) and 1003 controlling two
alternative data caches (1006, 1007) so that only one of the
caches is used at a time. Thus, one cache is always available

for compensating for defects and can be addressed over the multiplexers. The code cache leads over a multiplexer (1008) to the control unit (1009) of the processor, only one of which is used in this example. The integer units (1010, 1011, 1012) and the floating point units (1013, 1014, 1015) are controlled by the control unit. Two are always in operation at a time, with a third being available if one of the units fails. Two integer units are connected to the data bus (1018) over the bidirectional multiplexers (1016, 1017), and two floating point units are connected to the data bus (1018) over the bidirectional multiplexers (1019, 1020). The data bus is connected to the data cache over a bidirectional multiplexer (1021). Multiplexer 1003 is also designed to be bidirectional. The multiplexers are controlled according to the method described above. Multiplexers 1002, 1008, multiplexers 1003, 1021, multiplexers 1016, 1017 and multiplexers 1019, 1020 each form a dependent group.

Thus according to the method described here, the data cache and the code cache and one floating point unit and one integer unit can be replaced within the sample processor in the event of a fault.

Figure 11 illustrates a self-test sequence. Variable n, n ∈ (1, 2, ...) is the index of all algorithms, and defines the algorithm being used at that time. Each algorithm is in a first position (1101) which corresponds to Figure 7a, and a second mirrored position (1102) which corresponds to Figure 7b. Variable m, m ∈ (1, 2, ...) is the index of the test vectors to be calculated and compared. Within each algorithm, the quantity of test vectors is tested completely and then the algorithm is changed either from 1101 to 1102 or from 1102 to a new algorithm (n = n + 1). The test is terminated if n reaches the value after the last valid algorithm. If an error is found during calculation (1103, 1104) of the test vectors, error processing is performed as illustrated in detail in Figures 12a, b. After successful error processing, all the

algorithms are tested again to ensure that no new errors have occurred due to the correction, but in principle testing could also be continued at the site of the instantaneously active algorithm.

Two methods are proposed for error processing. Figure 12a shows the generation of new multiplexer states by way of a counter, where the variable v, $v \in (0, 1, \ldots$ (number of PAEs)) is the number of the defective PAE. If no PAE is defective, then $v = 0$. First v is increased so that the next PAE beginning with PAE 1, is marked as defective. Then the test that was failed is carried out again. If the test runs correctly, it is certain that PAE v is defective and v is written into a non-volatile read-write memory (e.g., 0903). If the test is failed again, v is increased until either the test runs correctly or v reaches the position after the last PAE and it is thus proven that the defective group of PAEs cannot be corrected, either because the error is located somewhere else (e.g., the bus system) or two or more PAEs are defective.

Figure 12b shows a less time-consuming option, where the look-up table described above is integrated. It receives as an input value indices m and n as well as the number of the comparator that found the error. This number is supplied by ErrorCHK (0706) over bus 0707. The look-up table supplies number v back to the defective PAE. Then the test that was failed is performed again. If the test runs correctly, it is certain that PAE v is defective and v is written into a non-volatile read-write memory (e.g., 0903). If the test is failed again, it is assumed that the error is irreparable. With most test algorithms, it is possible to determine the column of the defective PAE but not its row. Thus, the column of defective PAE v can be determined easily, but it is not known in which of the plurality of rows the defective PAE is located. Therefore, in such cases, the error processing of Figures 12a, b must be performed over all the rows that were involved in calculation of the test algorithm until the error has been

detected or all the rows have been tested and the error is
uncorrectable.

Figure 13 shows a possibility of performing a chip test during
IDLE cycles, i.e., cycles during which no program is being
executed because the system is waiting for an action (e.g., a
keyboard input). Such cycles are usually executed through
queues in the program code. It is readily possible when such a
waiting condition prevails to call up a test routine which
performs a test of the chip during the waiting time, although
a realtime capability is no longer possible in reaction to the
action awaited. Indices m and n known from Figure 11 are also
used in Figure 13 with the same meanings, but the indices are
stored in the data memory. All the relevant data in the array
is saved before calling up the test routine and is restored
again after execution of the test routine. When the test
routine is called up, the indices are first loaded from the
data memory. Then the corresponding algorithm with the
corresponding test vectors is executed. If the result is
faulty, error processing is performed according to Figures
12a, b. Otherwise, the indices are calculated again and
written back into the data memory. Then a test is performed to
determine whether an IDLE state still prevails, i.e., if
waiting for an action. If an IDLE state still prevails, the
test routine is started again, but this time a different
calculation is performed - according to the new calculation of
indices already performed. If there is no longer an IDLE
state, the program execution is continued in the normal way.

Figure 14 illustrates a possible modification of Figure 13,
where the test routine is called up directly by the
application program (call TEST_CHIP (m, n)). The test routine
is called up at certain suitable points in the algorithm. All
the relevant data in the array is first saved and is restored
again after execution of the test routine. Indices m, n are
also transferred directly on call-up. Within the TEST_CHIP
routine, algorithm n is executed with data m. The error test

is performed according to Figures 11 and 13. At the end of
TEST_CHIP, the indices are not recalculated, in contrast with
Figures 11 and 13. The return from the TEST_CHIP routine leads
to the position directly downstream from "call TEST_CHIP,"
5   which is comparable to the BASIC standard GOSUB ... RETURN.

Figure 15 illustrates a possible control of a look-up table.
The look-up table (1501) here is implemented as a ROM. Indices
m, n, i.e., the identifier of the test algorithm performed
10   instantaneously and the identifier of the current test data,
as well as the result (0707) of the ErrorCHK unit (0706) are
sent as addresses (1502) to the ROM. The resulting data (1503)
indicates the number of the defective PAE. This number is
transferred to the register (0108).
15

Figure 16 shows a fault-tolerant bus system. An additional bus
(1602) is assigned to a quantity of identical buses (1601).
The bus system has terminals in all four directions. The
terminals are connected to the buses over multiplexers (1603)
20   so that if one bus fails, its function is assumed by the
neighboring bus (below it in the horizontal or to the right in
the vertical). All the multiplexers connected to a certain bus
are addressed by the same control line, e.g., multiplexers
1603a which are connected to bus 1601a are controlled by
25   control line 1604a. Control lines 1604 are controlled by a
unit according to 0109 from Figure 1. Further control and
error processing are in principle the same as those described
with regard to the preceding figures.

30   Figure 17 shows the assignment of a chip-internal RAM or RAM
area (1701) to a group of cells (1702). Before executing the
test algorithms, the internal registers of the cells (1702)
are stored in the RAM or RAM area (1701). After execution of
the test algorithms, the data is written back into the
35   internal registers of the cells. The data is read and written
over multiplexers/gates (0103, 0104 and 0105). Thus, data
originating from a defective cell is written to the cell

connected as a replacement according to the position of the multiplexers. The sequence of the user algorithm is not impaired by the test method.

5   Figure 18 illustrates the system described in conjunction with Figure 17, but the data of the cells in the chip (1801) is written to an external memory (1802) or read out of the external memory.

10   Figure 19 illustrates the use of explicit test registers TestReg-n. A register TestReg-n (1903) used for the test algorithms is assigned to each internal register Reg-n of a cell (1902). The demultiplexers (the gate) (1901) selects which register is to be written to; the multiplexer (1904)
15   select which register is to be read from. The control of the (de)multiplexers (1901, 1904) is such that in normal operation, i.e., while the user algorithms are running, Reg-n (1902) is selected and TestReg-n (1903) is used in execution of the test algorithm. The circuit according to Figure 19 is
20   implemented for each relevant register within a cell. A disadvantage of this circuit is that the data of a defective cell is not available to the cell connected as a replacement. To overcome this disadvantage, it is proposed that an additional circuit based on the basic principle (0103, 0104
25   and 0105) according to the present invention be implemented by (de)multiplexers (1901, 1904). Thus, all the data can be made available to the replacement cell.

Definition of terms

User algorithm: user-specific program that runs on the chip.

5   Output multiplexer: circuit comparable to a relay which selects between multiple possible buses at the outputs of PAEs.

BIST: built-in self-test. Self-test implemented in an

10   integrated circuit, i.e., the TestMEM and all test functions are included in the circuit.

Booting: loading a basic program and setting the basic functions after a reset.

15

Bus: bundle of multiple individual signals/individual lines which together fulfil a certain transfer function (data bus, address bus ...).

20   EDST: external driven self-test. Self-test implemented in an integrated circuit where the TestMEM is connected outside the integrated circuit and parts of the control can also be accommodated outside the circuit.

25   Input multiplexer: circuit comparable to a relay which selects between multiple possible buses at the inputs of PAEs.

ErrorCHK: circuit implemented in an integrated unit which recognizes and localizes a fault during BIST or EDST.

30

HOST: a higher-order module or computer over an integrated circuit.

IDLE: state in which a processor or similar circuit waits for an action and does not perform any processing.

Look-up table: memory which returns a fixedly defined data

word of any desired width on the basis of an address; usually implemented as a ROM or ROM-like memory.

Neighbor bus: bus located directly next to another bus and having the same structure.

PAE: processing array element according to German Patent Application 196 51 075.9-53. Although a PAE in German Patent Application 196 51 075.9-53 is an arithmetic and logic unit, the term is used in more general terms in the present patent to represent any desired cell, i.e., arithmetic and logic units, state machines, memories, etc.

PAER: cell which is additionally implemented on the integrated circuit and can be used as a replacement for a defective cell of the same design.

RESET: returning an integrated circuit to a defined basic state. A reset occurs in general after applying a voltage (turning the circuit on).

Self-test: automatic test method implemented in an integrated unit.

Setpoint results: the results that must be supplied by the test algorithms to prove the correctness of the integrated circuit. There is an error if the setpoint results do not match the calculations.

Test algorithms: program that tests the integrated unit; it contains all the mathematical functions and circuits that are to be tested.

State machine: state machine for sequence control of complex operations; also called a sequencer.

Test algorithm: program for testing the chip or cells of a

structure.

Test data: the quantity of all test algorithms, test vectors and setpoint results.

5

TestMEM: memory in which the test data is stored.

Test routine: individual program section within the test algorithms.

10

Test vectors: data with which the test algorithms are executed.

Gates: switches that relay or block data.

15

Cell: self-contained module within an integrated circuit, e.g., arithmetic and logic units, state machines, memories.